THE UNIVERSITY OF ARIZONA

Midterm Exam – Thursday October 16, 9:30-10:45pm, 2000

COMPUTER SCIENCE
Foundations of Systems Programming
(Time allowed: 75 minutes)

SURNAME: ____________________________
GIVEN NAMES: _______________________
STUDENT ID NUMBER: __________________
SIGNATURE: __________________________

Answer all questions in the space provided. Write clearly and legibly, you will not get credit for illegible answers.

This is a closed-book test. You may not refer to any books or notes during this examination, and you may not use a calculator or any other electronic aid.

Print your name at the top of every page.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
</tr>
</tbody>
</table>
- 2 -

(a) Convert $1100110_2$ to decimal. Show your work.  

\[ 1100110_2 = 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 64 + 32 + 4 + 2 = 102_{10} \]

(b) Convert $21CA_{16}$ to decimal. Show your work.  

\[ 1100110_2 = 2 \cdot 16^3 + 1 \cdot 16^2 + 12 \cdot 16^1 + 10 \cdot 16^0 = 8192 + 256 + 192 + 10 = 8650_{10} \]

(c) Convert $237_{10}$ to binary. Show your work.  

\[
\begin{array}{c|c|c}
\text{Division} & \text{Quotient} & \text{Remainder} \\
237/2 & 118 & 1 \\
118/2 & 59 & 0 \\
59/2 & 29 & 1 \\
29/2 & 14 & 1 \\
14/2 & 7 & 0 \\
7/2 & 3 & 1 \\
3/2 & 1 & 1 \\
1/2 & 0 & 1 \\
\end{array}
\]

\[ 237_{10} = 11101101_2 \]

(d) Convert $-43_{10}$ into 2’s complement representation. Show your work.  

\[
\begin{align*}
43_{10} &= 0101011 \\
\text{not } 43_{10} &= 1010100 \\
(\text{not } 43_{10}) + 1 &= 1010101
\end{align*}
\]

CONTINUED
(e) Convert $-43_{10}$ into a BCD representation. The code for $'$—' is $1011_2$. [4 points]

$$-43_{10} = 1011\ 0100\ 0011_{BCD}$$
2. The *dot product* of two vectors

\[ [a_0, a_1, \ldots, a_{n-1}] \]

and

\[ [b_0, b_1, \ldots, b_{n-1}] \]

is given by

\[ a_0 * b_0 + a_1 * b_1 + \cdots + a_{n-1} * b_{n-1}. \]

A Java program to calculate this value from two vectors is:

```java
public class dot {
   public static int dotProduct (int a[], int b[], int n) {
      int v = 0;
      for(int i=0; i<n; i++) {
         v += a[i] * b[i];
      }
      return v;
   }

   public static void main (String args[]) {
      int a[] = {1,2,3,4};
      int b[] = {5,6,7,8};
      int v = dotProduct(a,b,4);
      System.out.println(v);
   }
}
```

Rewrite this program in MIPS assembly code. Include the `main` routine, the `dotProduct` routine, data declarations for the two vectors in the example above, and code to print out the final result.

The vectors are stored as one-dimensional arrays. The address of the first array is passed in register `$a0$, the address of the second vector in register `$a1$, and the length of the arrays in register `$a2$. The value of the dot product should be returned in register `$v0$. Be sure to follow the MIPS calling convention, set up a stack frame for all calls, and save `$fp$ and `$ra` on entry to all procedures.
.data
A:
.word 1
.word 2
.word 3
.word 4
B:
.word 5
.word 6
.word 7
.word 8
nl: .asciiz "\n"

.text
main:
  subu $sp,$sp, 24    # Allocate stack frame
  sw $fp, 0($sp)     # Save $fp in frame
  addu $fp, $sp, 24  # Set up $fp
  sw $ra, -20($fp)   # Save $ra

  la $a0, A
  la $a1, B
  li $a2, 4
  jal dotProduct

  move $a0,$v0
  li $v0,1
  syscall

  la $a0,nl
  li $v0,4
  syscall

  lw $ra, -20($fp)  # Restore $ra
  lw $fp, -24($fp)  # Restore $fp
  addu $sp, $sp, 24 # Pop stack frame
  jr $ra
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>dotProduct:</td>
<td>subu $sp,$sp, 24</td>
<td>Allocate stack frame</td>
</tr>
<tr>
<td></td>
<td>sw $fp, 0($sp)</td>
<td>Save $fp in frame</td>
</tr>
<tr>
<td></td>
<td>addu $fp, $sp, 24</td>
<td>Set up $fp</td>
</tr>
<tr>
<td></td>
<td>sw $ra, -20($fp)</td>
<td>Save $ra</td>
</tr>
<tr>
<td></td>
<td>li $v0, 0</td>
<td>Init return value</td>
</tr>
<tr>
<td>loop:</td>
<td>beqz $a2, done</td>
<td>Are we done?</td>
</tr>
<tr>
<td></td>
<td>lw $t1,($a0)</td>
<td>t1 = load A[i]</td>
</tr>
<tr>
<td></td>
<td>lw $t2,($a1)</td>
<td>t2 = load B[i]</td>
</tr>
<tr>
<td></td>
<td>mul $t3, $t1, $t2</td>
<td>t3 = A[i]*B[i]</td>
</tr>
<tr>
<td></td>
<td>add $v0,$v0,$t3</td>
<td>v0 = v0 + A[i]*B[i]</td>
</tr>
<tr>
<td></td>
<td>addu $a0,$a0,4</td>
<td>a0 = next elmt in A</td>
</tr>
<tr>
<td></td>
<td>addu $a1,$a1,4</td>
<td>a1 = next elmt in B</td>
</tr>
<tr>
<td></td>
<td>subu $a2,$a2,1</td>
<td>count down number of elmts left</td>
</tr>
<tr>
<td></td>
<td>b loop</td>
<td></td>
</tr>
<tr>
<td>done:</td>
<td>lw $ra, -20($fp)</td>
<td>Restore $ra</td>
</tr>
<tr>
<td></td>
<td>lw $fp, -24($fp)</td>
<td>Restore $fp</td>
</tr>
<tr>
<td></td>
<td>addu $sp, $sp, 24</td>
<td>Pop stack frame</td>
</tr>
<tr>
<td></td>
<td>jr $ra</td>
<td></td>
</tr>
</tbody>
</table>
3. (a) Fill in the table below:  

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>A and B</th>
<th>A nand B</th>
<th>A or B</th>
<th>A nor B</th>
<th>A xor B</th>
<th>A xnor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

(b) Fill in the table below to show how to do one bit addition with carry:  

<table>
<thead>
<tr>
<th>carry in</th>
<th>A</th>
<th>B</th>
<th>carry out</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
(c) Show how to compute $103_{10} - 66_{10}$ in a 2’s complement representation: [5 points]

\[
\begin{array}{ccc}
0110 0111_2 & \text{(103)} & \\
- 0100 0010_2 & \text{(66)} & \\
\hline
\downarrow & 2 \text{'s complement} & \\
0110 0111_2 & \text{(103)} & \\
+ 1011 1110_2 & \text{(-66)} & \\
\hline
= 0010 0101_2 & \text{(37)} &
\end{array}
\]

(d) Fill in the table below: [5 points]

<table>
<thead>
<tr>
<th>Binary</th>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>10110011</td>
<td>shift left logical 1</td>
<td>01100110</td>
</tr>
<tr>
<td>10110011</td>
<td>shift right logical 2</td>
<td>00101100</td>
</tr>
<tr>
<td>10110011</td>
<td>shift right arithmetic 2</td>
<td>11101100</td>
</tr>
<tr>
<td>10110011</td>
<td>rotate left 4</td>
<td>00111011</td>
</tr>
<tr>
<td>10110011</td>
<td>rotate right 2</td>
<td>11101100</td>
</tr>
</tbody>
</table>
(a) Using the MIPS Instruction Encoding sheet attached to this exam, convert the following MIPS assembly instruction into MIPS machine code (in hex). Show your work. [10 points]

\[ \text{sra } \$t0, \$s0, 5 \]

\[
\text{sra } \$t0, \$s0, 5 = \\
000000 0000 01000 10000 00101 000011 = \\
0000 0000 0000 1000 1000 0001 0100 0011 = \\
0x00088143
\]

(b) Using the MIPS Instruction Encoding sheet attached to this exam, convert the following MIPS machine code instruction into MIPS assembly code. Registers should be given in their symbolic form, e.g. \( \$t0 \) rather than \( \$8 \). Show your work. [10 points]

\( \text{0x27a50004} \)

\[
\text{0x27a50004} = \\
0010 0111 1010 0101 0000 0000 0000 0100 = \\
001001 1101 00101 0000000000000000100 = \\
\text{addiu } \$5, \$29, \$4 = \\
\text{addiu } \$a1, \$sp, \$4
\]
5. Consider the following code fragment:

```
.data
foo: .word 7
bar: .word 9
 .word 13

.text
main: la $t0, bar
lw $t1, 4($t0)
add $t1, $t1, $t1
slt $t2, $t1, 2
sw $t2, -4($t0)
addu $t0, $t0, 4
sw $t1, ($t0)
```

After this code is executed, what are the values of:

[20 points]

(a) bar's address+4
(b) 2 * 13 = 26
(c) 26 * 4 = 104
(d) 104
(e) 9
(f) 26